**Utilisez la bonne boucle pour répéter les tâches (for, while)**

En programmation, il y a des ensembles d'instructions à répéter plusieurs fois. Parfois, vous connaîtrez à l'avance le nombre de répétitions, d'autres fois non. Il est aussi possible que le nombre de fois n'importe pas, et que vous souhaitiez répéter le code jusqu'à atteindre une certaine condition. Pour tous ces cas, nous utiliserons des **boucles.**

**Utilisez la boucle for pour savoir "combien de fois ?"**

Supposons que vous ayez à faire embarquer 10 passagers, et que pour l'instant vous n'accordiez pas d'importance à l'ordre de leur embarquement. Utilisez une boucle  for  pour les embarquer un par un jusqu'à atteindre 10 :

*const* numberOfPassengers = 10;

for (*let* i = 0; i < numberOfPassengers; i++) {

*console*.log("Passager embarqué !");

}

Pour cette boucle  for, créez une variable d'indice  i  qui sert de compteur pour le nombre d'exécutions de la boucle. C'est pour cette raison qu'elle démarrera à zéro, car on n'a pas encore parcouru la boucle.

La deuxième commande dans les parenthèses  for  est la condition de poursuite de la boucle : dès qu'elle s'évalue comme  false, on quitte la boucle. Dans ce cas, vous souhaitez l'exécuter autant de fois qu'il y a de passagers, donc quand l'indice  i  atteint 10 (après 10 boucles), vous souhaitez l'arrêter, car il n'y a plus de passager.

La troisième commande demande à la boucle  for  d'incrémenter  i  (ajouter 1) à chaque exécution. C'est ce qui permet de suivre le nombre d'exécutions de la boucle.

JavaScript termine la boucle avant d'exécuter tout autre code, donc si vous deviez écrire ceci :

*const* numberOfPassengers = 10;

for (*let* i = 0; i < numberOfPassengers; i++) {

*console*.log("Passager embarqué !");

}

*console*.log("Tous les passagers sont embarqués !");

la dernière sortie de console s'imprimerait *après* l'embarquement des 10 passagers.

Le travail sur des indices c'est bien, mais si vous souhaitiez utiliser un tableau de passagers pour les embarquer dans un ordre précis ?

**Travaillez sur des tableaux :  for…of et for…in**

Dans beaucoup de cas, vous aurez un tableau et devrez le parcourir, pour faire quelque chose sur chaque élément. Il peut s'agir de mettre à jour les prix sur un menu, de traiter les données d'une base de données, ou d'embarquer des passagers dans l'ordre.

L'ancienne façon de parcourir un tableau dans une boucle était d'utiliser la boucle  for  vue précédemment avec sa propriété  length. Par exemple, avec un tableau appelé  passengers  :

for (*let* i = 0; i < passengers.length; i++) {

*console*.log("Passager embarqué !");

}

Bien que ceci fonctionne, il y a deux façons plus faciles de parcourir en boucle des tableaux (ou de faire une itération sur leurs éléments).

**La boucle for…in**

La boucle  for  …  in  est très comparable à l'exemple de boucle  for  normale, mais elle est plus facile à lire, et effectue tout le travail d'itération pour vous :

*const* passengers = [

"Will Alexander",

"Sarah Kate'",

"Audrey Simon",

"Tao Perkington"

]

for (*let* i in passengers) {

*console*.log("Embarquement du passager " + passengers[i]);

}

Comme dans l'exemple précédent,  i  démarre automatiquement à zéro, et s'incrémente à chaque boucle. Vous imprimez donc  passengers[0], puispassengers[1], puis  passengers[2], etc., jusqu'à terminer l'itération sur tous les passagers. Vous pouvez bien sûr imprimer chaque élément sur la console, car chacun est une chaîne contenant le nom du passager.

**La boucle for…of**

Pour les cas où l'indice précis d'un élément n'est pas nécessaire pendant l'itération, vous pouvez utiliser une boucle  for  …  of  :

En reprenant l'exemple précédent et en utilisant à la place une boucle  for  …  of, vous obtenez :

*const* passengers = [

"Will Alexander",

"Sarah Kate",

"Audrey Simon",

"Tao Perkington"

]

for (*let* passenger of passengers) {

*console*.log("Embarquement du passager " + passenger);

}

Ceci atteint exactement le même résultat, mais de façon plus lisible, car vous n'avez pas à vous inquiéter des indices et des tableaux : vous recevez simplement chaque élément dans l'ordre. C'est encore plus utile si le tableau est un peu plus complexe et contient par exemple des objets :

*const* passengers = [

{

name: "Will Alexander",

ticketNumber: 209542

},

{

name: "Sarah Kate",

ticketNumber: 169336

},

{

name: "Audrey Simon",

ticketNumber: 779042

},

{

name: "Tao Perkington",

ticketNumber: 703911

}

]

for (*let* passenger of passengers) {

*console*.log('Embarquement du passager ' + passenger.name + ' avec le ticket numéro ' + passenger.ticketNumber);

}

**Pratiquez : les boucles  for**

Voici un entraînement qui utilise des tableaux et des boucles for  .

Rendez-vous sur cet[éditeur CodePen](https://codepen.io/nicolaspatschkowski/pen/ZEGRGpG). Le composant épisode a été restructuré, et les épisodes sont maintenant importés depuis une base de données.

Maintenant, vous devez implémenter du code qui remettra à zéro l'information de visionnage pour tous les épisodes. Vous allez accomplir cette tâche en itérant le tableau  episode  en assignant la valeur false à la propriété  hasBeenWatched  de chaque épisode.

Le tableau  episodes  a déjà été déclaré pour vous.

1. Itérez le tableau  episodes  et assignez la valeur false à la propriété  hasBeenWatched  de chaque épisode, en utilisant une boucle for, for…in ou for…of.
2. Maintenant, faites en sorte que tous les épisodes soient considérés comme ayant été vus. Cela fonctionne t-il également ?

N'oubliez pas, votre code ira entre les commentaires "===" sur l'exercice.

**Solution :**

[Voici un nouveau CodePen avec une solution](https://codepen.io/nicolaspatschkowski/pen/dyorvOv) à l’exercice.

Ces types de boucles fonctionnent très bien pour un nombre fixe d'itérations. Mais vous ne pouvez pas toujours savoir combien de fois une boucle doit être exécutée.

**Utilisez la boucle  while  pour continuer jusqu'à ce qu'on dise d'arrêter**

Une boucle  while  vérifie si une condition est vraie. Si c'est le cas, la boucle se poursuit ; sinon elle s'arrête. En revenant à l'exemple de l'embarquement d'avion : supposons que vous ayez des variables représentant le nombre de sièges restants et le nombre de passagers restants. Vous souhaitez continuer l'embarquement jusqu'à l'épuisement du nombre de sièges ou de passagers :

*let* seatsLeft = 10;

*let* passengersStillToBoard = 8;

*let* passengersBoarded = 0;

while (seatsLeft > 0 && passengersStillToBoard > 0) {

passengersBoarded++; // un passager embarque

passengersStillToBoard--; // donc il y a un passager de moins à embarquer

seatsLeft--; // et un siège de moins

}

*console*.log(passengersBoarded); // imprime 8, car il y a 8 passagers pour 10 sièges

Cette boucle  while  poursuit son exécution jusqu'à ce que l'un des nombres  seatsLeft  et  passengersStillToBoard  atteigne zéro, et à ce point elle se termine.

**En résumé**

Dans ce chapitre, vous avez abordé deux façons de répéter les tâches :

* la boucle  for, pour un nombre d'itérations fixe ;
* la boucle  while, quand le nombre d'itérations nécessaire est inconnu.